IBDSim version 2.0

User manual

September 16, 2015

IBDSim is a computer package for the simulation of allelic and sequence data at multiple unliked loci under general isolation by distance models. It is based on a backward "generation by generation" coalescent algorithm allowing the consideration of various isolation by distance models on a lattice with deterministically varying deme size, migration rates and mutation rates. IBDSim can consider a large panel of subdivided population models representing subpopulations with or without a demic structure, the latter case being an approximation for continuous population. Many dispersal distributions can be considered as well as heterogeneities in space and time of the demographic parameters. Typical applications of our program include the study of the effect of various sampling, mutational and demographic factors on the pattern of genetic variation at different spatial scales and the production of test data sets to assess the influence of these factors on any inferential method available to analyze genotypic data for independent loci. IBDSim v2.0 now comes with a graphical user interface (GUI). The command-line program as well as its GUI runs on Windows (Xp, 7 and 8), MacOs X or most recent Linux distributions. We also provide the source code for the command-line version that can be compiled under any system using C++ ISO compiler. IBDSim v2.0 with its GUI is freely available on the website at http://raphael.leblois.free.fr/.

1 Requirements

1.1 Command-line executables and source compilation for various OS

The program `IBDSim` is available for download on the web at the address http://raphael.leblois.free.fr/ and is provided as a Windows executable as well as original source code.
Windows users can run the provided executables (`IBDSim.exe`, build with the MinGW port of the GNU Compiler Collection (GCC)).
Linux and Mac users should easily recompile the sources using `gcc` and the following command line

```
g++ -DNO_MODULES -o IBD_Sim lattice_sampling.cpp -O3
```

Various versions of the code have been compiled and run on PCs under Windows and Linux. Some preprocessor instructions were added to compile the code on these different systems. So this should essentially work on most Unix-based systems, including Mac OS X and previous versions (we have only tested it on MacOs X). When compiling with specific compilers (i.e. others than GCC) or specific IDE like wxDevC++ , one should sometimes manually edit/add/delete some "#include <library>" instructions in the first lines of the file "lattice_sampling.cpp".

1.2 Graphical user interface (GUI)

The program `IBDSim` is now available with a graphical user interface that should facilitate its usage by people not familiar with command-line tools. The GUI is available for download on the same web at the address http://raphael.leblois.free.fr/. `IBDSim`'s GUI is based on PyQt, a Python binding of the Qt framework. It runs on the three main operating systems: Microsoft Windows (Xp, 7 and 8), Apple MacOs X and most recent Linux distributions. It is distributed with a simple setup procedure adapted to each OS.

The GUI was build on two main ideas: (1) it is just an additional layer to the command-line executable that still works with a simple parameter text file containing all simulation settings and (2) it is self explanatory and thus easy to use. The GUI operates by producing a parameter file that could independently be run by the user with the command-line executable, so that it is easy to switch between the GUI and the command-line version. Further, the parameter file can be previewed in the GUI and filled step-by-step according to the options chosen by the user through the interface.

All keywords and option names are exactly the same in the GUI and in the parameter file. The screen outputs in the GUI are the same as when using the command-line version (cerr/cout). Finally, a "What's this" button displays help text bubbles for each option by just moving the pointer on a button or a tick box.

Using the GUI should thus be straightforward, all settings and keywords are the same than for the command-line version. For this reason, we do not mention the GUI further in this documentation.

1.3 Hardware

`IBDSim` should run on any reasonably recent computer and has limited memory needs for most reasonable settings. There is virtually no limitation or maxima for the parameter values concerning the lattice and subpopulation sizes, the sample size (i.e. number of individuals and loci), however high values will increase memory usage and decrease execution speed. Reasonable simulation times are usually obtained even with reasonably large lattices, population sizes and sample sizes (e.g few hours for 1000 data sets of 20 loci for 1000 individuals evolving on a 100×100 lattice with subpopulation sizes of 100 individuals). Note that considering heterogeneities in space will strongly increase computation time as well as memory usage, especially for large lattice sizes. Finally, simulating a large number of loci, e.g. for SNPs, increase memory usage. However, millions of loci can be easily simulated on any recent machine as it needs a few Gb's of RAM.

2 Your first `IBDSim` session: a simple example

In this section, we describe a simple example, so that you can directly get in touch with the software while starting to read in detail the whole documentation.
For this example, we consider a demographic model of Isolation by distance with 20×20 populations, each population being a panmictic unit with 30 haploid individuals (i.e. Isolation by distance with demic structure). Dispersal is simulated using a stepping stone migration model (i.e. migration only occurs between adjacent populations) with a migration rate of 0.05. Ten data files are simulated with multilocus genotypes of 20 haploid individuals sampled from 4 populations taken on a 2×2 square in the center of the lattice . For all individuals, we simulate 3 microsatellite loci evolving under a strict stepwise mutation model with a mutation rate of 0.0001 and allele sizes comprised between 1 and 200 repeats.

The content of the setting file `IbdSettings_firstSessionExample.txt` with all the above described parameters is the following:

```
%%%%% SIMULATION PARAMETERS %%%%%%%%%%%%
Data_File_Name=Example
Run_Number=10

%%%%% MARKERS PARAMETERS %%%%%%%%%%%%%%%
Locus_Number=3
Mutation_Rate=0.001
Mutation_Model=SMM
Allelic_Lower_Bound=1
Allelic_Upper_Bound=200

%%%%%% SAMPLE PARAMETERS %%%%%%%%%%%%%%%
Sample_SizeX=2
Sample_SizeY=2
Min_Sample_CoordinateX=9
Min_Sample_CoordinateY=12
Ind_Per_Pop_Sampled=5

%%%%%%%% DEMOGRAPHIC PARAMETERS %%%%%%%%%%
Ind_Per_Pop=30
Lattice_SizeX=20
Lattice_SizeY=20
Dispersal_Distribution=stepping_stone
Total_Emigration_Rate=0.05
```

First, copy the provided `IbdSettings_firstSessionExample.txt` into an empty folder and rename it `IbdSettings.txt` (Be careful to respect capital letters under Linux, it does not matter for Windows or macOS). Make the `IBDSim` executable accessible from this folder by whatever mean suitable for your operating system (either copy the executable file or launch it from a distant folder e.g. `../ExecutableFolder/IBDSim`). Launch the executable. Wait for the completion of the computation which should last only a few seconds. The output files generated by `IBDSim` are (i) 10 text files named Example_1.txt to Example_10.txt; (ii) a file named `simul_pars.txt` with a summary of the input settings and some information about dispersal distributions; and (iii) a file named `migraine.txt` which is a template input file for analyzing the simulated data sets of Example_1.txt to Example_10.txt using `Migraine` (Rousset & Leblois, 2007, 2012; Leblois *et al.*, 2014, see section 4.5 for details about the interaction between `IBDSim` and `Migraine`). This file is so far relatively basic and will work only for simple models, e.g. 1D and 2D IBD, and a single population with past variation in population size (see help for details about those anlayses) with a single type of marker (i.e. homogeneous mutational processes among markers).

Each of the simulated data sets is written as a `Genepop` input file and has the following format (example for output file Example_1.txt, see section 4.2.3 for more details on `Genepop` file format):

```
This file has been generated by the IBDSim program.
locus1_SMM
locus2_SMM
locus3_SMM
pop
9 12 , 040 043 138
9 12 , 040 056 138
9 12 , 043 054 138
9 12 , 043 043 138
9 12 , 044 043 136
pop
9 13 , 048 056 138
9 13 , 044 043 134
9 13 , 048 056 138
9 13 , 040 045 138
9 13 , 050 047 137
pop
10 12 , 043 043 136
10 12 , 043 056 138
10 12 , 039 045 134
10 12 , 043 056 138
10 12 , 043 043 138
pop
10 13 , 042 047 134
10 13 , 043 047 138
10 13 , 043 043 129
10 13 , 051 047 129
10 13 , 043 045 137
```

3 Principle of the simulation algorithm and implemented models

3.1 Coalescent algorithm

The `IBDSim` program is based on the backward-in-time coalescent approach, which is well known for allowing the development of efficient simulation tools (Hudson, 1993). Such an approach allows the generation of large genotypic data sets considering complex migration schemes including those with heterogeneities in space and time of the demographic parameters. Moreover, because our program allows various deme size and migration rates, it can sim-

ulate genotypic data under both a model of subdivided population with discrete subpopulations and a model corresponding to a large pseudo-continuous population with any intermediate situation.

For neutral genes, the coalescent process depends solely on the demographic history of the population and is independent from the mutational processes. So we first generate the genealogy of the sampled genes going backward in time and then simulate mutations starting from the top of the coalescent tree (i.e. MRCA: Most Recent Common Ancestor) and adding them independently along all branches of the tree.

The coalescent algorithm used to build the genealogical tree is not based on the large-N approximation of the n-coalescent theory (Kingman, 1982; Nordborg, 2007). It is rather an exact algorithm for which coalescence and migration events are considered generation by generation up to the common ancestor of the sample. The idea of tracing lineages back in time, generation by generation, is fundamental in the coalescence theory, and is well described in Nordborg (2007). Such a generation-by-generation algorithm leads to less efficient simulations in terms of computation time than those based on the n-coalescent theory. However, this algorithm is much more flexible when complex demographic and dispersal features are considered. The generation-by-generation algorithm that gives the coalescent tree for a sample of n genes evolving under IBD has been detailed in Leblois *et al.* (2003, 2004, 2006) and the main ideas underlying the global algorithm are summarized in Leblois *et al.* (2009). The algorithm and the program used in this study were checked at every step during their elaboration by comparing simulated values of probabilities of identity of two genes under models of isolation by distance on finite lattices with their exact analytically computed values (e.g. Malécot (1975) for the lattice model) adaptated to different mutation models following Rousset (1996).

3.2 Migration models

Specifying a dispersal model proceeds in several steps. One must first specify a forward dispersal distribution, describing emigration probabilities to different distances (`Dispersal_Distribution` setting). Next one must specify how habitat boundary effects are handled (`Edge_Effects` setting). These will typically affect the immigration probability in each deme; for example, demes at the boundary may receive fewer immigrants than more central demes. However, even the immigration probability in central demes may be reduced compared to the emigration probability of the unbounded forward distribution. In some cases one may wish to have an exact control of the immigration probability, for example to assess the performance of estimators of

this probability. For that purpose, the `Immigration_Control` option allows one to further control the construction of backward dispersal distributions from forward ones.

3.2.1 Forward dispersal distributions

Biologically realistic dispersal functions often have a high kurtosis (Endler, 1977; Kot *et al.*, 1996). However, commonly used discrete probability distributions are not the most appropriate ones for isolation by distance because they imply that high kurtosis can be achieved only by assuming a low dispersal probability, i.e. that most offspring reproduce exactly where their parents reproduced (Rousset, 2000). Therefore, we have implemented two less well-known families of dispersal distributions that allow high kurtosis and high migration rates: the Pareto and the Sichel. The better known geometric distribution, with the stepping stone as a special case, is also implemented. For two-dimensional models, we assume that dispersal is independent in each direction, so that $f_{dx,dy} = f_{dx} \cdot f_{dy}$.

The first family of distributions are truncated variants of the discrete Pareto, or Zeta, distribution (see e.g. Patil & Joshi, 1968) with the probability of moving k steps (for $-Dist\_max \leq k \leq Dist\_max$, $k \neq 0$) in one direction being of the form:

$$f_k = \frac{M}{2 \cdot |k|^n} \tag{1}$$

with parameters M and n, controlling the total dispersal rate and the kurtosis, respectively.

The second family of dispersal distributions is obtained as mixtures of convolutions of stepping stone steps and is a convenient way to model discrete distributions with various forms (Chesson & Lee, 2005). As detailed in that paper, the Sichel mixture is described by three parameters, ξ, ω and γ . Parameterization of the Sichel mixture distribution is not trivial but details on each parameter and formulas to compute various moments of the distribution as well as its kernel are given in Chesson & Lee (2005). Both the full three-parameter distribution, and the long-tailed variant of this family obtained in the limit case $\omega \to 0$, $\xi \to \infty$ with $\omega\xi \to \kappa$ are implemented. In the latter case the two parameters γ and κ then describes a family of distributions which are Gaussian-looking at short distances but have tails proportional to $r^{-2\gamma-1}$ for distance r. The values of γ and κ can be chosen so as to achieve some given second moment (σ) and kurtosis. For more details on the Sichel distribution parametrization, see Watts *et al.* (2007) and Chesson & Lee (2005).

Finally, we also considered geometric dispersal distributions for which the probability of moving k steps (for$-Dist_max \leq k \leq Dist_max$, $k \neq 0$) in one direction is:

$$f_k = \frac{M}{2} g^{|k|-1}, \tag{2}$$

with m controlling the total emigration rate and g the shape of the distribution. The stepping stone dispersal is the limit of the geometric distribution with $g \to 0$, and the finite island model of dispersal is the limit of the geometric distribution with $g \to 1$.

The above dispersal distributions can be selected by values of `Dispersal_Distribution` setting listed below, where in each case we describe the additional parameters to be specified. Details on the default values and syntax for the additional parameter keywords are given p.36.

`Dispersal_Distribution=b` or `SteppingStone`: custom stepping stone distribution with total emigration rate set in the input file by the keyword `Total_Emigration_Rate`.

`Dispersal_Distribution=g` or `Geometric`: custom geometric distribution with total emigration rate and shape set in the input file by the keywords `Total_Emigration_Rate` and `Geometric_Shape`respectively. Note that high kurtosis cannot be achieved with a geometric distribution without small emigration rates.

`Dispersal_Distribution=p` or `Pareto`: custom truncated Pareto distribution with parameters M and n set in the input file by the keywords `Total_Emigration_Rate` and `Pareto_Shape`, respectively.

`Dispersal_Distribution=s` or `Sichel`: custom Sichel mixture distribution with parameters ξ, ω and γ set in the input file by the keywords `Sichel_Gamma`, `Sichel_Xi`, `Sichel_Omega`. Some parameter values which gives biologically realistic dispersal distributions can be found in Watts *et al.* (2007). Detailed description of this distribution is given in Chesson & Lee (2005).

For ease of reproducibility of published results, specific cases of the above distributions (for specific parameter values), and a maximal dispersal distance $Dist\_max$, in lattice steps, can also be selected through the `Dispersal_Distribution` setting (see option `Total_Range_Dispersal` p.34 for more details on dispersal ranges). These more specific options are

1. `Dispersal_Distribution=0`: truncated Pareto distribution (see p.9) with $\sigma^2 = 4$ and $Dist\_max = 15$, and parameters $M = 0.3$ and $n = 2.51829$ for $k > 2$, and with $f_1 = f_{-1} = 0.06$ and $f_2 = f_{-2} = 0.03$ for $k \leq 2$.

2. `Dispersal_Distribution=1` stepping stone distribution with total emigration rate $M = 2/3$.

3. `Dispersal_Distribution=2` truncated Pareto distribution with $\sigma^2 = 1$ and $Dist\_max = 49$, and parameters $M = 0.599985$ and $n = 3.79809435$.

4. `Dispersal_Distribution=3` truncated Pareto distribution with $\sigma^2 = 100$ and $Dist\_max = 48$, and parameters $M = 0.6$ and $n = 1.246085$.

5. `Dispersal_Distribution=4` truncated Pareto distribution especially designed for lattices with one non-empty node every second (see option `Void_Nodes` p.35) with $\sigma^2 = 1$ and $Dist\_max = 48$, and parameters $M = 0.824095$ and $n = 4.1078739681$.

6. `Dispersal_Distribution=5` truncated Pareto distribution especially designed for lattices with one non-empty node every third (see option `Void_Nodes` p.35) with $\sigma^2 = 1$ and $Dist\_max = 48$, and parameters $M = 0.913378$ and $n = 4.43153111547$.

7. `Dispersal_Distribution=6` truncated Pareto distribution with $\sigma^2 = 20$ and $Dist\_max = 48$, and parameters $M = 0.719326$ and $n = 2.0334337244$.

8. `Dispersal_Distribution=7` truncated Pareto distribution with $\sigma^2 = 10$ and $Dist\_max = 49$, and parameters $M = 0.702504$ and $n = 2.313010658$.

9. `Dispersal_Distribution=8` truncated Pareto distribution especially designed for lattices with one non-empty node every third (see option `Void_Nodes` p.35) with $\sigma^2 = 4$ and $Dist\_max = 48$, and parameters $M = 0.678842$ and $n = 4.1598694692$.

10. `Dispersal_Distribution=9` truncated Pareto distribution with $\sigma^2 = 4$ and $Dist\_max = 48$, and parameters $M = 0.700013$ and $n = 2.74376568$.

11. `Dispersal_Distribution=a` stepping stone distribution with total emigration rate $M = 1/3$.

3.2.2 Spatial repartition of individuals and habitat shape

`IBDSim` considers isolation by distance models on a lattice, and not on continuous space, strictly speaking (but see Robledo-Arnuncio & Rousset (2010) for details on continuous and lattice models). `IBDSim` can either consider models with demic structure, i.e. where each lattice node is a panmictic population of size N individuals; or pseudo-continuous models, where each lattice node

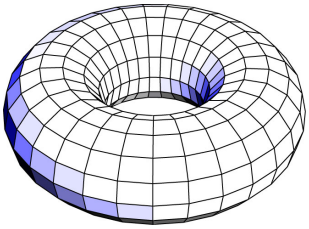

Figure 1: Graphical representation of a torus

is a single individual.

Mathematical analyzes of isolation by distance models usually consider lattice models without edge effect (i.e. on a circle or a torus in one and two dimensions respectively, Fig. 1) to have complete homogeneity in space, which strongly facilitates analytical developments. However, as torus or circle models are not generally realistic, we implemented various edge effects in `IBDSim`:

- no edges: the lattice is represented on a circle or a torus for a one or a two-dimensional model respectively;
- reflective boundaries: the lattice is represented on a line or plane and trajectories of dispersal events going outside the lattice are reflected on edges as light is reflected on a mirror;
- absorbing boundaries: the lattice is represented on a line or plane and all individuals that emigrate (forward) out of the habitat are lost (i.e. the probability mass of coming (backward) outside the lattice is equally shared on all other movements inside the lattice).

3.2.3 Effects of edges, heterogeneous density and barrier to gene flow on backward distributions

We used the "backward" dispersal distribution in the coalescent algorithm because the position of the parental gene is determined knowing the position of its descendant gene (remember that dispersal is gametic and thus involves haploid entities). This "backward" function is computed using $f_{dx,dy}$, the forward dispersal density function describing where descendants go. In the simplest case, considering that density is homogeneous in space and that there is no edge effects (i.e. `Edge_Effects=Circular`, the population is represented by a circle or torus), backward dispersal functions are equal to forward dispersal functions, so that $b_{dx} = f_{dx}$ for one-dimension models and

$b_{dx,dy} = f_{dx,dy} = f_{dx} \cdot f_{dy}$ for two-dimensional habitat with independent dispersal in each dimension.

However, when density is not totally homogeneous in space, backward and forward dispersal differ. In this case, each lattice node has a backward distribution that depends on the density of each surrounding node. Those surrounding nodes correspond to all locations from which genes could have come in one generation (forward in time). Since those nodes are occupied by different numbers of individuals and because nodes occupied by more individuals contribute potentially more to the number of immigrants that reach a given node, we have to weight each term of the backward dispersal distribution by the number of individuals of the node where immigrants come from. Then for any node $\mathbf{z}_1$ the probability $b_{\mathbf{z}_1,\mathbf{z}_2}$ that a gene is immigrant from $\mathbf{z}_2$ is equal to

$$b_{\mathbf{z}_1,\mathbf{z}_2} = \frac{N_{\mathbf{z}_2} f_{\mathbf{z}_1-\mathbf{z}_2}}{\sum N_{\mathbf{z}} f_{\mathbf{z}_1-\mathbf{z}}} \tag{3}$$

where the sum is over all possible nodes $\mathbf{z}$ that are defined inside the lattice, non-empty (as implied by $N_{\mathbf{z}}$), and within a maximum $Dist\_max$ distance if any such distance is considered in the specification of the forward distribution (option `Total_Range_Dispersal` p.34).

Backward and forward distributions also differs when there is a barrier to gene flow (option `ForwardBarrier`, p. 38). In such case, with similar notations than those used above, backward dispersal distribution for any node of the lattice is equal to

$$b_{\mathbf{z}_1,\mathbf{z}_2} = \frac{\delta m_{\text{barrier},\mathbf{z}_1,\mathbf{z}_2} f_{\mathbf{z}_1-\mathbf{z}_2}}{\sum \delta m_{\text{barrier},\mathbf{z}_1,\mathbf{z}} f_{\mathbf{z}_1-\mathbf{z}}} \tag{4}$$

where the sum is over all possible nodes $\mathbf{z}$ that are defined inside the lattice (and within a maximum $Dist\_max$, see above) and $\delta m_{\text{barrier},\mathbf{z}_1,\mathbf{z}_2}$ is equal to the barrier crossing rate (option `Barrier_Crossing_Rate`, p. 39) if the straight line $(\underline{z}_1, \underline{z}_2)$ cross the barrier, or equals 1 otherwise.

3.2.4 Immigration control

Without further specification, the immigration rate in any lattice node is only remotely related to the `Total_Emigration_Rate` setting of each forward dispersal distribution, first as the result of edge effects (with absorbing boundaries, the immigration rate will be lower than the forward emigration rate as some emigrants are lost outside the habitat), second (in two dimensions) because `Total_Emigration_Rate` only gives the one-dimensional emigration rate and, without edge effects, the two-dimensional non-dispersal rate will rather be the square of 1−`Total_Emigration_Rate`.

To allow a finer control of the immigration rate, the following option is available for the geometric dispersal distributions :
`Immigration_Control=1DproductWithoutm0` allows one to exactly control the maximal immigration rate in a node, in both linear and two-dimensional habitats. Because of edge effects, the immigration rate will typically differ among nodes (with absorbing boundaries, more immigrants will come into central nodes that into peripheral ones). With this option, the forward dispersal probability is adjusted such that the maximal immigration probability in a deme has the given value `Total_Emigration_Rate` , and that the deme of origin of immigrants is chosen according to relative forward dispersal probabilities, identical for all parental demes. Therefore, the backward immigration probability in a focal deme can be written in the form

$$\frac{f(x,y)M/G}{\sum_{k\neq d} f(x,y)M/G + (1-M)} \tag{5}$$

where M is the `Total_Emigration_Rate`, G is the maximum value, over all demes each taken as the focal one d, of $\sum_{k\neq d} f(x,y)$, and $\sum_{k\neq d}(.)$ denotes a sum over source demes k distinct from the focal deme d, as in eq. 3. For the maximizing focal deme, the denominator is 1, the immigration probability from a distinct deme is $Mf(x,y)/\sum_{k\neq d} f(x,y)$, and the non-immigration probability is exactly $1-M$. Note that backward dispersal is not independent in each dimension in this case; rather, either an individual does not disperse, or it moves independently in each dimension. This second step also allows for an additional probability of no dispersal, and 1-`Total_Emigration_Rate` is the total non-immigration probability implied by these two steps.

The default `Immigration_control` option applicable to all families of dispersal distributions are: `Immigration_Control=Simple1Dproduct`, which is the default defined by eq. 3.

3.2.5 Pre- and postdispersal sampling, life cycle

For all simulations, the life cycle is divided into five steps: (i) each individual gives birth to a great number of gametes, and dies; (ii) gametes undergo the effect of mutations; (iii) gametes disperse; (iv) diploid individuals are formed, if necessary, by considering Hardy?Weinberg equilibrium within each deme, and (v) competition brings back the number of adults in each deme toN, with $N = 1$ for the continuous model and $N > 1$ for the demic model.

By default, samples simulated by `IBDSim` are th us composed of individuals sampled after the last dispersal step in the life cycle. However, since `IBDSim` V2.0, one can also simulate samples taken before the last dispersal step (see section 4.2.5). This feature has notably been implemented to infer

dispersal using methods based on the consideration of pre- and post-dispersal samples, such as the one described in Vitalis (2002).

3.3 Mutation models

`IBDSim` can simulate allelic (e.g. microsatellite) and DNA sequence data. Currently 12 mutation models are implemented in `IBDSim` and all options for the genetic markers are given in 4.2.2. The implemented models are the following:

3.3.1 Allelic data

1. `IAM`: the infinite allele model (IAM, Kimura & Crow, 1964) in which each mutation gives rise to a new allele;

2. `KAM`: the K-allele model (KAM, Crow & Kimura, 1970) in which a mutation changes the initial allelic state into one of $K-1$ other possible states;

3. `SMM`: the strict stepwise mutation model (SMM, Ohta & Kimura, 1973), much applied to microsatellite markers, where each mutation adds or removes a repeated unit to the mutated allele;

4. `TPM`: the two phase model (TPM, Di Rienzo *et al.*, 1994), where each mutation adds or removes X repeated units to the mutated allele. With a probability p, X is equal to 1 and with a probability of $1-p$ X is randomly chosen from a geometric distribution with variance v, implying a gain or a loss of more than one repeated unit;

5. `GSM`: the generalized stepwise model (GSM, e.g. Pritchard *et al.*, 1999), similar to the TPM but there is only one "phase" of geometric loss or gain of X repeated units with variance v. The GSM is a TPM with $p = 0$;

3.3.2 DNA Sequence data

6. `SNP`: Single Nucleotide Polymorphisms denote specific nucleotide positions (distributed over a larger sequence, e.g. chromosome) known to exhibit bi-alletic polymorphism. To simulate SNP loci, we proceeded following the algorithm proposed by Hudson (2002) (cf -s1 option in the program `ms`, Hudson 2002). Briefly, the genealogy at a given ?locus is simulated until the most recent common ancestor (see 3.1) and a single mutation event is put at random on one branch of the genealogy (the branch being chosen with a probability proportional to its length relatively to the

total gene tree length). This way of simulating SNPs is discussed in Cornuet *et al.* (2014, Supplementary Appendix S1) Another way to simulate SNPs is to simulate a KAM with 2 states, and use the following options `Min_Allele_Number=2`, which sets the minimum number of alleles that a locus needs to have to be incorporated into the simulated data sets to 2, or `Polymorphic_Loci_Only=True` that is equivalent, in combination with `Max_Mutation_Number=1`, which sets the maximum number of mutations that can occur on a locus to be incorporated into the simulated data sets. Finally, note that the option `Minor_Allele_Frequency=0.X` can be used to set a lower bound on the frequency of the lesser abundant allele at the particular locus.

7. `ISM`: the infinite sites model (ISM,Kimura, 1969), in which each mutation in the coalescent tree gives rise to a unique segregating nucleotide (or a previously unmutated site of the sequence). `IBDSim` thus simulates a sequence whose size is equal to the number of mutations occurring in the coalescent tree;

8. `JC69`: the Jukes-Cantor model (JC69,Jukes & Cantor, 1969), is a nucleotide substitution model where the equilibrium frequencies of the purine bases (`A` and `G`) and the pyrimidine bases (`C` and `T`) are assumed to be in an equal proportion (*i.e.* $\pi_A = \pi_G = \pi_C = \pi_T = \frac{1}{4}$) and the rate of substitution from any given base to the other three bases is the same irrespective of the ancestral or the mutant base;

9. `K80 / K2P`: the two parameter Kimura model (K80,Kimura, 1980), is a simple generalisation of the JC69 model where the nucleotide substitutions are categorized as either transitions (*i.e.* `A`↔`G`, `C`↔`T`) or as transversions (*i.e.* `A`↔`C`, `A`↔`T`, `G`↔`C`,`G`↔`T`). Real data typically contains some form of a transition-transversion bias which can be explained by the relative rates of these two categories of substitutions. This bias can be specified in `IBDSim` using the ratio of the rate of transition substitutions over transversion substitutions for every substitution (*see also* 4.2.2). In the absence of a transition-transversion bias, (ex. the JC69 model), this value is by default $\frac{1}{2}$, as a given ancestral base (*say* G) has always three possible mutant states; one of which is always a transition substitution (*in this case* A) and the other two are transversion substitutions (*i.e.* C or T).

10. `F81`: the Felsenstein'81 model (F81,Felsenstein, 1981), is another generalisation of the JC69 model where we do not distinguish between transitions and transversions and instead allow the equilibrium base frequencies to vary (*i.e.* $\pi_A \neq \pi_G \neq \pi_C \neq \pi_T$). In this case the transition-transversion

ratio needn't be specified as it is the same as that of the JC69 model (*i.e.* $\frac{1}{2}$);

11. `HKY85`: the Hasegawa, Kishino and Yano model (HKY85, Hasegawa *et al.*, 1985), integrates aspects of both the K80/K2P and F81 models by respectively letting the equilibrium base frequencies to be variable and the transition-transversion bias to be specified;

12. `TN93`: the Tamura-Nei model (TN93, Tamura & Nei, 1993) is the most general of the substitution models available in `IBDSim` which further generalizes the HKY85 model by allowing an additional bias to be introduced between purine transitions and pyrimidine transitions. This can be specified in terms of the ratio of the rate of `A`↔`G` substitutions over that of `C`↔`T` substitutions for every substitution (*see also* 4.2.2).

4 All `IBDSim` options

4.1 Input file format

`IBDSim` reads a single generic text file (ASCII), whose default name is `"IbdSettings.txt"`, which must be in the same folder as the application (Be careful to respect capital letters under Linux, it does not matter for Windows or macOS). The file is read at the beginning of each execution and allows one to control all options of `IBDSim`. It contains lines of the form `keyword=value(s)` or of the form `keyword=keyword1,keyword2,keyword3`, where value(s) and keyword(s) can take various formats as described below. Note that all booleans will be evaluated using a convenient procedure allowing the following symbols/keywords: `T, True, Yes, Y, F, False, No, N`. All setting options and their default values are explained in details in the following subsections. The default name of the settings file is `IbdSettings.txt` but you can change this through the command line: running `IBDSim` using `IBDSim SettingsFile=mysettings.txt` will make the program read `mysettings.txt` rather than `IbdSettings.txt` (Note that complete path can be included in the filename). Again, be careful to respect capital letters under Linux, it does not matter for Windows or macOS.

Note that the settings file format has changed between version 1.4 and 2.0, and they are not fully compatible. `IBDSim` V2.0 and later can read the old format of the setting file but will not consider any demographic change in time. The main difference is that the old version needed all keywords and values to be specified even if they were not used, whereas the new format

allows one to only specify parameters that are (1) effectively used by the program and (2) different from the default values. Some keywords have also been changed, and obsolete keywords are no longer documented.

4.1.1 A simple example

To start with, here is an example of the settings file (in the new format) for simulating a pseudo-continuous IBD population at constant time and space:

```
%%%%% SIMULATION PARAMETERS %%%%%%%%%%%%
Data_File_Name=ContPop
Run_Number=10

%%%%% MARKERS PARAMETER S%%%%%%%%%%%%%%%
Locus_Number=5
Mutation_Rate=0.0005
Mutation_Model=SMM
Allelic_Upper_Bound=200

%%%%%%%% DEMOGRAPHIC OPTIONS %%%%%%%%%%%%%
%% LATTICE
Lattice_SizeX=500
Lattice_SizeY=500
Ind_Per_Pop=1

%% SAMPLE
Sample_SizeX=15
Sample_SizeY=15
Min_Sample_CoordinateX=200
Min_Sample_CoordinateY=200
Ind_Per_Pop_Sampled=1

%% DISPERSAL
Dispersal_Distribution=9
```

4.1.2 Multimaker simulations

IBDSim V2.0 can now simulate multiple markers (allelic and/or sequence data) using a single settings file. This example simulates multiple markers with the same demographic settings as the previous example with some additional settings for DNA sequence loci (and for a fewer number of simulations):

```
%%%%% SIMULATION PARAMETERS %%%%%%%%%%%%
Data_File_Name=ContPop
Run_Number=2
```

```
%%%%% MARKERS PARAMETERS %%%%%%%%%%%%%%%
Locus_Number = 2, 1, 2, 1
Mutation_Rate = 0.001, 0.002, 0.003, 0.004
Mutation_Model = ISM, KAM, SMM, TN93
Allelic_Upper_Bound = NA, 20, 50, NA

%% SEQUENCE SPECIFIC SETTINGS
Sequence_Size = 50
Transition_Transversion_ratio = 0.7
Transition1_Transition2_ratio = 1.2
Equilibrium_Frequencies = 0.1 0.4 0.3 0.2

%%%%%% OUTPUT FILE FORMAT OPTIONS %%%%%%%
Genepop=T
Genepop_no_coord=F
Group_All_Samples=F
Geneland=F
Migrate=F
Nexus_file_format = Haplotypes_only

%%%%%%%% DEMOGRAPHIC OPTIONS %%%%%%%%%%%%%
%% LATTICE
Lattice_SizeX=500
Lattice_SizeY=500
Ind_Per_Pop=1

%% SAMPLE
Sample_SizeX=15
Sample_SizeY=15
Min_Sample_CoordinateX=200
Min_Sample_CoordinateY=200
Ind_Per_Pop_Sampled=1

%% DISPERSAL
Dispersal_Distribution=9
```

4.1.3 A complete example

Here, we present a complete (i.e with almost all keywords) settings file in the IBDSim V2.0 format. It mainly extends the previous example to two temporal demographic changes and some spatial heterogeneity in density (i.e. a high density zone):

```
%%%%% SIMULATION PARAMETERS %%%%%%%%%%%%
Data_File_Name=TestIBDSim
Genepop_File_Extension=.txt
Run_Number=10
```

```
Random_Seeds=87144630
Pause=Final

%%%%% MARKERS PARAMETERS %%%%%%%%%%%%%%%
Locus_Number = 2, 1, 2, 1
Mutation_Rate = 0.001, 0.001, 0.001, 0.002
Mutation_Model = ISM, KAM, SMM, TN93
Variable_Mutation_Rate = F, F, T, F
Polymorphic_Loci_Only = T, F, F, T

Allelic_Lower_Bound = NA, 3, 5, NA
Allelic_Upper_Bound = NA, 20, 50, NA
Allelic_State_MRCA = NA, 8, 6, NA
Repeated_motif_size = NA, 1, 1, NA
SMM_Probability_In_TPM = NA, 0.01, 0.2, NA
Geometric_Variance_In_TPM = NA, 2.6, 3.5, NA
Geometric_Variance_In_GSM = NA, 0.56, 0.93, NA

Ploidy=Diploid

%% SEQUENCE SPECIFIC SETTINGS
MRCA_Sequence = AGCTAGCTAGCT
%Note, Sequence_Size is redundant  information here! %
Sequence_Size = 12
Transition_Transversion_ratio = 0.7
Transition1_Transition2_ratio = 1.2
Equilibrium_Frequencies = 0.1 0.4 0.3 0.2

%%%%%% OUTPUT FILE FORMAT OPTIONS %%%%%%%
Genepop=T
Genepop_no_coord=F
Group_All_Samples=F
Geneland=F
Migrate=F
Migrate_Letter=F
Nexus_file_format = Haplotypes_only

%%%%%% VARIOUS COMPUTATION OPTION S%%%%%%%
%The code below can be specified in a single line %
DiagnosticTables=Iterative_Identity_Probability,Hexp,Fis,Seq_stats
DiagnosticTables=Prob_Id_Matrix,Effective_Dispersal,Iterative_statistics

%%%%%%%% DEMOGRAPHIC OPTIONS %%%%%%%%%%%%%
%% LATTICE
Lattice_Boundaries=absorbing
Total_Range_Dispersal=F

Lattice_SizeX=100
Lattice_SizeY=50
```

```
Ind_Per_Pop=100
Void_Nodes=1

Specific_Density_Design=false
Zone=T
Void_Nodes_Zone=1
Ind_Per_Pop_Zone=50
Min_Zone_CoordinateX=5
Max_Zone_CoordinateX=15
Min_Zone_CoordinateY=5
Max_Zone_CoordinateY=15

%% SAMPLE
%% classical squared sample design:
Sample_SizeX=10
Sample_SizeY=10
Min_Sample_CoordinateX=5
Min_Sample_CoordinateY=5
%%   OR !!  exclusive settings !! specific sample design:
%Sample_Coordinates_X=3 4 8 9 10 11 12 13 17 18
%Sample_Coordinates_Y=1 6 9 12 15 16 17 18 19 20

Ind_Per_Pop_Sampled=1

%% DISPERSAL
Dispersal_Distribution=1
Immigration_Control=Simple1DProduct
Total_Emigration_Rate=0.1
Dist_max=48
Pareto_Shape=2.16574
Geometric_Shape=0.75
Sichel_Gamma=-2.15
Sichel_Xi=20.72
Sichel_Omega=-1

%% DISPERSAL ZONE
Dispersal_Distribution_zone=1
Total_Emigration_Rate_zone=0.1
Dist_max_zone=48
Pareto_Shape_zone=2.16574
Geometric_Shape_zone=0.75
Sichel_Gamma_zone=-2.15
Sichel_Xi_zone=20.72
Sichel_Omega_zone=-1


%% CONTINUOUS CHANGE IN DENSITY
ContinuousDemeSizeVariation=F
```

```
%% CONTINUOUS CHANGE IN LATTICE SIZE
ContinuousLatticeSizeVariation=F

%% BARRIER TO GENE FLOW
ForwardBarrier=T
BackwardBarrier=F
x1_Barrier=50
x2_Barrier=50
y1_Barrier=1
y2_Barrier=50
Barrier_Crossing_Rate=0.2


%% FIRST DEMOGRAPHIC CHANGE
NewDemographicPhaseAt=50
Ind_Per_Pop=20
Lattice_SizeX=50
Lattice_SizeY=50
Random_Translation=true

Void_Nodes=2
Zone=false
Void_Nodes_Zone=1
Ind_Per_Pop_Zone=1

Dispersal_Distribution=1

ContinuousDemeSizeVariation=Exponential

%% SECOND DEMOGRAPHIC CHANGE
NewDemographicPhaseAt=200
Ind_Per_Pop=1
Lattice_SizeY=10
Random_Translation=true
%%%%%% EndOfSettings %%%%%%%%
```

4.2 Description of all options

All values or keywords specified before the brackets [ ] in this documentation will correspond to the default values/keywords implemented in `IBDSim`. Other possible inputs are given between brackets [ ] after the default values.

e.g. `Some_Param=default_val` [`other_val1` or `other_val1` or...]

4.2.1 Simulation parameters

All options in this category are quite straightforward to understand:

- `Data_File_Name=gp` is the generic file name for the simulated data sets. This generic file name will be incremented with the number of the run. Example: simulated data file number 56 will be named here `gp_56`.

- `.txt_extension=true` [or `false`] tells `IBDSim` to add or not to add a `.txt` extension to each simulated file. Example: if set to true, simulated data file number 56 will be named here `gp_56.txt`.

- `Genepop_File_extension=""` [or `. followed by any characters` , e.g. `.txt`, `.gp`] tells `IBDSim` to add an extension to each simulated file. Example: if set to `.gp`, simulated data file number 56 will be named here `gp_56.gp`.

- `Run_Number=10` tells `IBDSim` to run a given number of iterations, i.e. a given number of simulated data sets, here 10.

- `Random_seeds=14071789` are the (concatenated) seeds for the random number generator. Different runs with precisely the same parameter values and same seeds will give exactly the same results.

- `Pause=Default` [or `Final` or `OnError`] tells `IBDSim` to stop the program and wait for a user intervention (Press a key) to resume. The `Default` behavior is that `IBDSim` will never stop, `OnError` means that `IBDSim` will pause on errors, and `Final` means that `IBDSim` will pause at the end of the run, letting the terminal/command window open until the user presses any key. The recommended settings for batch simulations is `Pause=Default` or `Pause=Final`.

4.2.2 Genetic marker parameters

In this section most of the options deal with the parametrization of the genetic markers simulated by `IBDSim` (see 3.3). When simulating multiple markers and/or loci, they can be expanded into a vector of values which corresponds to either the number of specified markers or the total number of loci (see 4.1.2 for an example). In case a multiple valued parameter doesn't correspond to either of the formats `IBDSim` generates an error. These potentially vector valued parameters are documented below with the prefixed `"*"` sign.

e.g. `*Some_Param=default_val` [`other_val1` or `other_val1` or...]

Important: When simulating mutiple markers/loci, specifying a single value for a potentially multiple valued parameter tells `IBDSim` to accept that

value for all the simulated markers/loci. Similarly, in the absence of a user-specified value, IBDSim uses the default value. For simulating mutiple markers/loci, the user needs to specify at least the total number of loci or the total number of markers, otherwise IBDSim throws an error when it encounters a vector valued parameter setting. If you have the courage and patience, you can also specify parameters locus-wise. In 4.1.2, Mutation_Rate was specified using a marker-wise format; its equivalent in an expanded locus-wise format would be:

```
Mutation_Rate = 0.001, 0.001, 0.002, 0.003, 0.003, 0.004
```

- *Locus_Number=10 is either the number of loci to simulate per data set or a vector of values specifying how the loci need to be treated, individually (i.e. locus-wise) or marker-wise.
- *Mutation_Model=KAM [or IAM or SMM or TPM or GSM or SNP or ISM or JC69 or K80 or K2P or F81 or HKY85 or TN93] sets the mutation model for all loci unless specified marker-wise/locus-wise. See 3.3 for a general description of all the models.
- *Mutation_Rate=0.0005 is the mutation rate per generation for all simulated loci unless specified marker-wise/locus-wise.
- *Variable_Mutation_Rate=False [or True] tells IBDSim to simulate a constant or a variable mutation rate among loci. If a variable mutation rate is chosen, IBDSim will automatically draw random mutation rates for each locus in a Gamma distribution with parameters (shape and scale) being (2, Mutation_Rate/2) so that the mean mutation rate across loci is equal to value specified by Mutation_Rate.
- *Polymorphic_Loci_Only=False [or True] . If True, this option tells IBDSim to only consider polymorphic loci for the simulated data set. If Polymorphic_Loci_Only=True, IBDSim will keep simulating the current locus until it finds a minimum of 2 alleles.
- *Min_Allele_Number=1 sets the minimum number of alleles that a locus needs to have to be incorporated into the simulated data sets. If Min_Allele_Number>1, IBDSim will keep simulating the current locus until it finds a minimum of Min_Allele_Number alleles for each data set. Specifying a value of 2 is thus equivalent to Polymorphic_Loci_Only=True. Note that this option over-rides that of Polymorphic_Loci_Only.

- `*Max_Mutation_Number=2,147,483,647` sets the maximum number of mutations that can occur on the genealogy of a locus to be incorporated into the simulated data sets. This option can be used to simulate SNPs by setting `Max_Mutation_Number=1` in combination with `Min_Allele_Number=2` or `Polymorphic_Loci_Only=True`.

- `*Minor_Allele_Frequency=0` [$0 < . < 0.5$] is meaningful only for the SNP model. It sets the a lower bound on the frequency of the lesser abundant allele at the particular locus.

- `*Allelic_Lower_Bound=1` sets the lowest possible allelic state for the mutation models KAM, SMM, TPM and GSM.

- `*Allelic_Upper_Bound=10` sets the largest possible allelic state for the mutation models KAM, SMM, TPM and GSM (*e.g.* for a KAM, the number of possible allelic states K is then given by K=`Allelic_Upper_Bound - Allelic_Lower_Bound` +1).

- `*Allelic_State_MRCA=0` [or any integer between `Allelic_Lower_Bound` and `Allelic_Upper_Bound`] sets the allelic state of the MRCA for the mutation models KAM, SMM, TPM and GSM. When `Allelic_State_MRCA=0`, the allelic state is uniformly drawn between `Allelic_Lower_Bound` and `Allelic_Upper_Bound` independantly for each locus and data file. If `Allelic_State_MRCA=`X ($X \neq 0$), the allelic state of the MRCA is always fixed to X.

- `*Repeated_motif_size=1` set the length of the repeated motif for mutation models adapted to microsatellite loci. With `Repeated_motif_size=2,4`, `IBDSim` will simulate microsatellite loci with repeated motives composed of di- and tetra-nucleotides.

- `*SMM_Probability_In_TPM=0.8` is a specific option for the TPM model (where each mutation adds or removes X repeated units to the mutated allele). With a probability `SMM_Probability_In_TPM`, X is equal to 1 and with a probability of 1−`SMM_Probability_In_TPM`, X is randomly chosen from a geometric distribution (see next option), implying a gain or a loss of more than one repeated unit.

- `*Geometric_Variance_In_TPM=10` is a specific option for the TPM model which sets the variance of the geometric distribution when a mutation (*i.e.* X number of repeated units) for the TPM does not follow a strict SMM (*i.e.* when $X = 1$).

- `*Geometric_Variance_In_GSM=0.36` is a specific option for the GSM model which is similar to the TPM but where there is only one phase of geometric loss or gain of X repeated units. It sets the variance of the geometric distribution from which X is drawn. The GSM model is a TPM model with `SMM_Probability_In_TPM`=0.

- `Ploidy=Haploid` [or `Diploid`] set the level of all the loci/marker simulated using the same settings file. Note that the diploid model assumes gametic dispersal. It is therefore equivalent to a haploid model; only the format of the output file is diploid.
  The parameters below apply only to the sequence substitution models JC69, K80/K2P, F81, HKY85 and TN93 unless specified otherwise.

- `Transition_Transversion_ratio=0.5` (also known as a Ti/Tv bias) is the user-specified probability of a transition substitution against a transversion substitution and applies only to the K80/K2P, HKY85 and TN93 models. (*see* pg. 16). In the absence of a user-specified Ti/Tv value `IBDSim` uses 0.5 (the value for the JC69 and F81 models). This value means that for every nucleotidic substitution there are twice as many possible transversions than transitions.

- `Transition1_Transition2_ratio=1` is the user-specified probability of a purine (A or G) transition substitution against a pyrimidine (C or T) transition substitution (*see* pg. 16). This option only applies to the TN93 model.

- `Equilibrium_Frequencies=0.2 0.3 0.2 0.3` is the default vector of population base frequencies of A, G, C and T respectively. The frequencies should sum up to unity. This option only applies to the F81, HKY85 and TN93 models where the equilibrium base frequencies can differ from each other.

- `Sequence_Size=50` defines the size of the sequence to be simulated by `IBDSim` by randomly drawing nucleotides from the equilibrium base frequencies which depend on the chosen substitution model. If you have also specified the `MRCA_Sequence`, then this option will be considered as redundant by `IBDSim`.

- `MRCA_Sequence` is the user-specified sequence of nucleotides for the MRCA. If this option has not been specified by the user, it is generated randomly by drawing from the discrete distribution of equilibrium base frequencies which depend on the substitution model.

4.2.3 Data set format options

These options set the different data file formats to be generated for each data set simulated by `IBDSim`. These data files can be then analyzed by other programs such as `Genepop`, `Geneland`, `Migraine MIGRATE` , `DIYABC`, `Structure`, or any other program than can read one of the three following formats.

- `Genepop=true` [or `false`] tells `IBDSim` whether to write each data file in the classical and widely used `Genepop` format (actually the extended input file format of `Genepop` v.4; Rousset, 2008). Here is an example:

```
example of input file for Genepop
loc1
loc2
pop
0.56 8.67, 0101 0102
pop
1.67 8.5, 0101 0102
```

where each line represents the genotype of one individual at different loci, and groups of individuals (samples from different populations) are separated by `pop` statements. For each population sample, the values before the coma of the last individual indicates geographic coordinates of the populations. When the option `Group_All_Samples` is set to `True`, no `pop` indicator is placed between geographic samples. See the `Genepop` documentation for details and examples.

- `Geneland=true` [or `false`] tells `IBDSim` whether to write each data file in the `Geneland` format. `Genelandneeds` two file, one with the genotypes, here is an example:

```
198 000 358 362 141 141 179 000 208 224 243 243 278 284 86 88 120 124
200 202 000 358 141 141 183 183 218 224 237 243 276 278 88 88 120 124
198 000 358 362 141 141 179 000 208 224 243 243 278 284 86 88 120 124
200 202 000 358 141 141 183 183 218 224 237 243 276 278 88 88 120 124
```

where each line represents the genotype of one individual at different loci (here 9 loci)

The second file is the coordinate of the individuals, with one line per individual and two columns for the two coordinates. It looks like that :

```
25.6 745.2
54.1 827.8
32.6 654.2
45.3 863.9
```

See the Geneland documentation for details and examples.

The Geneland genotype file format can also be used as in put file for *Structure*, with the option ONEROWPERIND. See the *Structure* documentation for details and examples.

- Migrate=false [or true] tells IBDSim to write or not each data file in the MIGRATE file format (Beerli & Felsenstein, 2001). The generic input file format for MIGRATE is the following, where < token > in angle-brackets is obligatory, {token} in curly brackets is obligatory for some:

```
<Nb of pops> <nb of loci> {delimiter between alleles}
<Nber of individuals> <title for pop 0-79>
<Individual 1 10-10> <data>
<Individual 2 10-10> <data>
....
<Nber of individuals> <title for pop 0-79>
<Individual 1 10-10> <data>
<Individual 2 10-10> <data>
....
```

Here is an example of Migrate input data file with microsatellite loci:

```
2 3 . Rana lessonae: Seeruecken versus Tal
2 Riedtli near G\"undelhart-H\"orhausen
0 42.45 37.31 18.18
0 42.45 37.33 18.16
4 Tal near Steckborn
1 43.46 33.37 18.18
1 44.46 33.35 19.18
1 44.46 35.? 18.18
1 43.42 35.31 20.18
```

- Migrate_Letter=False [or True] tells IBDSim to write or not each data file in the special MIGRATE file format with letters a..z or 1-digit numbers 0..9 representing alleles (Beerli & Felsenstein, 2001). Note that this option is limited to a maximum number of alleles of 36 in IBDSim. Here is an example of such format with allelic states represented as letters (e.g for allozymes):

```
2 11 Migration rates between two Turkish frog populations
3 Akcapinar (between Marmaris and Adana)
PB1058 ee bb ab bb bb aa aa bb ?? cc aa
PB1059 ee bb ab bb bb aa aa bb bb cc aa
PB1060 ee bb b? bb ab aa aa bb bb cc aa
2 Ezine (between Selcuk and Dardanelles)
```

```
PB16843 ee bb ab bb aa aa aa cc bb cc aa
PB16844 ee bb bb bb ab aa aa cc bb cc aa
```

- `Nexus_file_format=False` [`Haplotypes_and_Individuals` or `Haplotypes_only`] tells `IBDSim` to write or not each data file in the NEXUS file format. For each run, the output can further generate a single NEXUS file which consists of either the haplotypes or both the simulated genotypes and the haplotypes in separate NEXUS files. The names of the file containing only the haplotypes and that containing all the genotypes are appended to the generic file name (the `Data_File_Name` setting, see 4.2) by `Haplotypes` and by `AllIndividuals` respectively. Before terminating with a `.nex` file extension, each NEXUS file is further appended by #Run-#Locus, where #Run is specified by the `Run_Number` setting (4.2) and #Locus is the order of the sequential loci specified by the `Locus_Number` and `Mutation_model` settings.
  Below is an example of a NEXUS file generated by `IBDSim` (more details on this format can be found here). However, note that the only difference from the NEXUS format is the mention of the MRCA as the first sequence of the data matrix denoted by the *Anc* prefix. The MRCA information also needs to be taken into account by the *ntax* specifier

```
#NEXUS

begin data;
dimensions ntax=9 nchar=12;
format datatype=dna symbols="ACTG";
matrix
Anc         AGCTAGCTAGCT
001         AGGGAGCCACCT
002         AGCAAGATCGCT
003         AGGGAGCCACCC
004         AGCAAGATCGCA
005         AGAGAACCACCT
006         AGCAAGATCGGT
007         ACCAAGATCGCT
008         ATCGAGCTATCG
;
end;
```

4.2.4 Various computational options

IBDSim v.2 introduces a new way of setting various computation options using a single keyword `DiagnosticTables` followed by one specific keyword

for each computation option. However, the old settings still work, and were of the form *Keyword*=`False` [or `True`]. For more details on all possible output files generated by `IBDSim`, go to section 4.3.

- `DiagnosticTables= [Hexp,Fis,Iterative_Statistics,Seq_stats Iterative_Identity_Probability,Prob_Id_Matrix, Allelic_Variance,Effective_Dispersal`
  tells `IBDSim` whether to compute various statistics on the genotypes data files or on the runs and to report them in the output files `various_statistics.txt` for mean values among all runs, and `Iterative_Statistics.txt` for all values for each locus and each simulated data file. These options are not compatible with all sample designs (see `Sample_Coordinates_X` option). See the details of each option below for more details:

- `Hexp` tells `IBDSim` to compute the expected Nei's heterozygosity and to report it in the output files `various_statistics.txt` and `Iterative_Statistics.txt`.

- `Fis` tells `IBDSim` to compute the Fis and the observed heterozygosity and to report it in the output files `various_statistics.txt` and `Iterative_Statistics.txt`.

- `Seq_stats` tells `IBDSim` to calculate the number of segregating sites ot the number of nucleotide positions which exhibit polymorphism compared to the MRCA sequence. It also calculates the average number of pairwise differences and the empirical Ti/Tv ration (see pg. 16). This information is written to `Iterative_Statistics.txt` for all the simulated loci.

- `Allelic_Variance` tells `IBDSim` to compute the variance in allelic size as well as the $M-$statistic of Garza & Williamson (2001) and to report them in the output files `various_statistics.txt` and `Iterative_Statistics.txt`. Those two statistics are designed for microsatellite markers only, and are not compatible with IAM and KAM mutation models.

- `Iterative_Statistics` tells `IBDSim` to compute, for each simulated data file and for each locus, the different statistics described in `DiagnosticTables` and to report them in the output file `Iterative_Statistics.txt` (see section 4.3 for details about the `Iterative_Statistics.txt` file format). When this option is set to `False`, `IBDSim` only reports average values for a run.

- `Iterative_Identity_Probability` tells `IBDSim` to compute, for each simulated data file, identity probabilities for the pairs of sampled genes and to report them in the output file `Iterative_IdProb.txt`. This option

is essentially implemented to plot the evolution of identity probabilities through the run to check the program against analytical expectations. It is not compatible with a specific sample design (see `Sample_Coordinate_X` option).

- `Prob_Id_Matrix` tells `IBDSim` to write an output file named `Matrix_IdProb.txt` with the matrix of identity probabilities as a function of the place of the sampled genes on the lattice. This setting is not compatible with a specific sample design (see `Sample_Coordinate_X` option).

- `Effective_Dispersal` tells `IBDSim` to compute the empirical effective backward dispersal distribution computed from all backward dispersal events that occurred during the multilocus simulation for each data set. Empirical dispersal distances are computed considering the habitat as a plane, even if the simulation settings actually considers a torus. Discrepancies between theoretical and empirical dispersal distributions are thus expected for all edge effects, especially for small size lattices and/or large maximal dispersal distances. Because of these discrepancies, the interpretation of the realized backward dispersal distribution given the settings specified for the forward distribution is often difficult and troubling. This empirical distribution is then written in a file named `EmpDisp_`*CurrentDataFileName*`.txt`. At the end of the file various statistics (mean, σ^2, kurtosis and skewness) are computed on the whole distribution and on the semi distribution (axial values). Another file, named `EmpImmigRate_CurrentDataFileName.txt` is also produced with empirical immigration rates for each node of the lattice. Finally, mean values along the whole simulation (i.e. for all data sets), are summarized in two files named `MeanEmpDisp.txt` and `MeanempImmigRate.txt`

4.2.5 Sample parameters

In this section all settings for the sample configuration are specified.

These sample parameters have to be compatible with some of the time-independent and time-dependent demographic options detailed in the next two sections.

As detailed in section 3.2.5, IBDSim by default (and in versions $<$ V2.0) simulates postdispersal samples (i.e. genes are samples after the last dispersal event in the life cycle). However, since `IBDSim`V2.0, one can also simulate samples taken before the last dispersal step in the life cycle. Sample options and output files by default concerns only postdispersal sampling, they may sometimes contain the keyword 'postdisp' (e.g. 'Iterative_IdProb_postdisp.txt', 'Various_Statistics_postdisp.txt',...). On the contrary, all keywords, options,

or output files concerning predispersal sampling always contain the keyword 'predisp'.

All options for simulating predispersal and postdispersal samples are listed below.

- `Sample_SizeX=1` is the axial number of sampled nodes in dimension X.
- `Predisp_Sample_SizeX=1` same as above but for predispersal sampling.
- `Sample_SizeY=1` is the axial number of sampled nodes in dimension Y.
- `Predisp_Sample_SizeY=1` same as above but for predispersal sampling.
- `Min_Sample_CoordinateX=1` is the coordinate of the most left sampled node in dimension X.
- `Predisp_Min_Sample_CoordinateX=1` same as above but for predispersal sampling.
- `Min_Sample_CoordinateY=1` is the coordinate of the most left sampled node in dimension Y.
- `Predisp_Min_Sample_CoordinateY=1` same as above but for predispersal sampling.
- `Sample_Coordinates_X=2 5 7 9 10 12 21 34 56` [No Default values] is a list of specific X coordinates for a user defined specific sample design. This option is not compatible with `DiagnosticTables`.
- `Predisp_Sample_Coordinates_X=2 5 7 9 10 12 21 34 56` [No Default values] same as above but for predispersal sampling.
- `Sample_Coordinates_Y=4 8 15 17 20 26 34 50 56` [No Default values] is a list of specific Y coordinates for a user defined specific sample design. Its length must be the same as the length of `Sample_Coordinates_X`. This option is not compatible with `DiagnosticTables`.
- `Predisp_Sample_Coordinates_Y=4 8 15 17 20 26 34 50 56` [No Default values] same as above but for predispersal sampling.
- `Ind_Per_Pop_Sampled=1` is the number of individuals sampled on each lattice node (i.e. "subpopulation" or individual for the continuous population model) within the sampled area.

- `Predisp_Ind_Per_Pop_Sampled=1` same as above but for predispersal sampling.
- `Void_Sample_Node=1` controls whether every node within the previously designed sampling area is sampled or not. With a value of 1 `IBDSim` will sample all nodes within the sampling area, with a value of 2 `IBDSim` will only sample one node every second, etc...
- `Predisp_Void_Sample_Node=1` same as above but for predispersal sampling.
- `Group_all_Samples=False [ or True]` tells IBDSim to group or not all geographic samples so that, when set to True, the samples appears in the `Genepop`file format as a single population (i.e. without `pop` indicators).
- `Genepop_No_Coord=False [ or True]` tells IBDSim not to write geographic coordinates in the `Genepop`file format. They are replaced by the number of the individual.

4.2.6 Time-independent demographic parameters

In this section all settings for the demographic part of the model that are independent of time are specified. These time-independent parameters correspond to demographic settings kept at fixed values during a whole simulation run. They sometimes have to be compatible with the time dependent demographic options detailed in the next section.

- `Lattice_Boundaries` or `Edge_Effects=Absorbing` [or `Circular` or `Reflecting`] sets the habitat (i.e. lattice) boundaries type to be considered for the entire simulation. Habitat boundaries cannot be changed through time. See section 3.2.2 for details on the different possible habitat boundaries implemented in `IBDSim`.
- `Min_Zone_CoordinateX=1` is the lowest coordinate (left border) in dimension X of the "zone" (i.e. a portion of the lattice where demographic parameters are different from the rest of the lattice, see option `Zone` p. 35).
- `Max_Zone_CoordinateX=1` is the highest coordinate (right border) in dimension X of the "zone".
- `Min_Zone_CoordinateY=1` is the lowest coordinate (bottom border) in dimension Y of the "zone".

- `Max_Zone_CoordinateY=1` is the highest coordinate (top border) in dimension Y of the "zone".

- `Random_Translation=False` [or `True`] tells `IBDSim` where to place the smaller lattice on the larger one, after a change in time of the lattice size. If true it will be randomly placed on the larger surface, if false it is placed on the left most bottom corner of the larger lattice.

- `Specific_Density_Design=false` [or `true`] tells `IBDSim` to consider (1) homogeneous density on the lattice if set to false; Or (2) a user specific density configuration of the lattice where each node of the lattice has a number of individuals (i.e. deme size) specified in a file named DensityMatrix.txt, if set to true. The default name of the specific density file is can be changed this through the command line: running `IBDSim` using `IBDSim DensityFile=myDensityFile.txt` will make the program read `myDensityFile.txt` rather than `DensityMatrix.txt` (Note that complete path can be included in the filename).
  The format of DensityMatrix.txt is a matrix with X coordinates in rows and Y coordinates in column. The file begin with coordinate X=0 and Y=0 in the upper left corner, so that the density matrix specified in DensityMatrix.txt is an upside down image of the lattice. With `Specific_Density_Design=true` , it is better to use specific `Sample_Coordinates` with sampled nodes corresponding to lattice nodes where density is greater than 0, to avoid bad behavior of the program.

- `Total_Range_Dispersal=True` [or `False`] tells `IBDSim` to keep maximum dispersal distances from the specific settings of the chosen distribution (or by the option `Dist_max` p.36) rather than constrained by lattice size (i.e. individuals can disperse up to $Dist\_max$ steps, where $Dist\_max$ is set according to each dispersal distribution settings rather than limited by lattice size only, see section 3.2 for more details on dispersal distributions).

- `Immigration_Control=Simple1Dproduct` [or `1DproductWithoutm0`] sets the methods for the computation of backward dispersal probabilities from the 1 dimensional forward distributions specified in the settings. More details are given in p.13 and further discussion on backward dispersal distributions can be found in Rousset & Leblois (2012) . Users who want to use `Migraine` or any other inference method for the number of migrants between populations Nm should be careful with these settings and are advised to read Rousset & Leblois (2012).

4.2.7 Time dependent demographic parameters

This section concerns all demographic parameters that can change through time. This part is very different from previous version of `IBDSim` (i.e. v<2.0). With the new settings, `IBDSim` will consider a constant model until it finds the keyword `NewDemographicPhaseAt=`X followed by a set of new demographic parameters corresponding to the parameters changing between the old and the new demographic phase at generation X. The number of demographic changes is no longer limited, and now concerns all parameters described below without any limitation. The model will be constant through time if there is no keyword `NewDemographicPhaseAt=`X in the settings file or if these keywords are not followed by any new values for any parameters.

- `NewDemographicPhaseAt=`X [No Default value] tells `IBDSim` to consider a new demographic phase starting at generation X, with new demographic parameters specified below this keyword in the settings file. This keyword is not required to specify the first demographic phase starting at generation 0.

- `Ind_Per_Pop=1` is number of individual per lattice node that `IBDSim` will consider for the current demographic phase specified above (or not for constant time models). It also correspond to the density in number of individuals per lattice node.

- `Lattice_SizeX=10` is the lattice size in the first dimension X.

- `Lattice_SizeY=1` is the lattice size in the second dimension Y.

- `Void_Nodes=1` is a setting to consider that a given proportion of lattice nodes are empty (i.e. they do not carry any individuals of the population). It has been implemented to decrease density without changing dispersal functions in Leblois *et al.* (2004). It can generaly be used to consider low densities (e.g less than one individual per lattice node) without changes in total lattice surface and dispersal distributions. With a value of 1, `IBDSim` will consider that all lattice node have individuals on them. With a value of 2, `IBDSim` will consider that one every two nodes is empty and can not receive any individual during simulation.

- `Zone=False [ OR True ]` tells `IBDSim` if there are heterogeneities in space in the density/subpopulation sizes by considering a special demographic "zone" (i.e. a portion of the lattice where demographic parameters are different from the rest of the lattice).

- `Void_Nodes_Zone=1` is the equivalent of the option `Void_Nodes` but for the specific demographic "zone".
- `Ind_Per_Pop_Zone=1` is number of individual per lattice node in the special demographic "zone" if there is one. In other words, it is the density in number of individuals per lattice node in the special demographic "zone".
- `Dispersal_Distribution=SteppingStone` [or `Geometric` or `Pareto` or `Sichel` or `b` or `g` or `p` or `s` or `a` or `0` or `1` or `2` or `3` or `4` or `5` or `6` or `7` or `8` or `9`] Its argument is a character, either a letter or a number, referring to one of the implemented dispersal distributions. This option tells `IBDSim` to consider one of the custom or preset dispersal distribution on the time interval considered. Detailed descriptions of all implemented dispersal distribution and parameters of these distributions are given in section 3.2.1. The following options are thus only described here in terms of keywords and default values.
- `Total_Emigration_Rate=0.5` is the total emigration rate (i.e. probability to disperse) for the stepping stone model, the truncated Pareto distribution and the geometric distribution.
- `Pareto_Shape=5` is the shape parameter value of the custom truncated Pareto distribution (see the description of truncated Pareto distribution on p.9).
- `Geometric_Shape=0.5` is the shape parameter value of the geometric distribution (see p.10).
- `Sichel_Gamma=-2.15` is the first parameter of the Sichel distribution (see the complete Sichel distribution description p.9).
- `Sichel_Xi=100` is the second parameter of the Sichel distribution.
- `Sichel_Omega=-1` is the third parameter of the Sichel distribution.
- `Dist_max=` [Default values depends on the dispersal distribution chosen] is the maximum distance moved at each generation, or the bound of the dispersal distribution, in lattice steps, for the geometric distribution, the custom Pareto and the Sichel distribution, when the option `Total_Range_Dispersal=True`. If `Total_Range_Dispersal=False`, `Dist_max=X` has no effect and the maximum distance moved is the size of the lattice in each dimension.

- `Dispersal_Distribution_zone=SteppingStone` corresponds to the option `Dispersal_Distribution` described above but for the specific zone if defined. It must have a maximal distance (`Dist_max`) less or equal to the one of the distribution used outside the zone.

- `Total_Emigration_Rate_zone=0.5`corresponds to the option `Total_Emigration_Rate` described above but for the specific zone if defined.

- `Pareto_Shape_zone=5`corresponds to the option `Pareto_Shape` described above but for the specific zone if defined.

- `Geometric_Shape_zone=0.5`corresponds to the option `Geometric_Shape` described above but for the specific zone if defined.

- `Dist_max_zone=` corresponds to the option `Dist_max` described above but for the specific zone if defined. It must be $<$ to `Dist_max`.

- `Continuous_Deme_Size_Variation=None` [or `Linear` or `Exponential` or `Logistic` or `False`] tells `IBDSim` to consider (1) time constant density on the lattice if set to false; Or (2) a linear, exponential or logistic continuous change in density bewteen Gn_X and Gn_{X+1}. By a continuous change in density we mean a continuous change in deme size, i.e. the number of individuals at each lattice node. Note that this option requires a later `New_Demographic_Phase_At` to specify the ancient density at Gn_{X+1}.

  Using `Continuous_Deme_Size_Variation=Linear`, the population size between Gn_X and Gn_{X+1} is given by $N(Gn) = N_{Gn_X} + (N_{Gn_{X+1}} - N_{Gn_{X+1}})\frac{Gn-Gn_X}{Gn_{X+1}-Gn_X}$ where the time Gn, in generations, is considered backward in time.

  Using `Continuous_Deme_Size_Variation=Exponential`, the population size between Gn_X and Gn_{X+1} is given by $N(Gn) = N_{Gn_X}$ $\exp\left(\log(\frac{N_{GN_{X+1}}}{N_{Gn_X}})\frac{Gn-Gn_X}{Gn_{X+1}-Gn_X}\right)$ where the time Gn, in generations,is considered backward in time.

  Using `Continuous_Deme_Size_Variation=Logistic`, the population size between Gn_X and Gn_{X+1} is given by $N(Gn) = \frac{N_{Gn_{X+1}}}{1+\frac{N_{Gn_{X+1}}-N_{Gn_X}}{N_{Gn_{X+1}}}\exp(-\texttt{DensLogisticGrowthRate}\cdot(Gn-Gn_X))}$ where the time Gn, in generations,is considered backward in time.

- `Dens_Logistic_Growth_Rate=0` is used with `Continuous_Deme_Size_Variation=Logistic` to specify the growth rate of the logistic.

- `Continuous_Lattice_Size_Variation=None` [or `Linear` or `Exponential` or `Logistic` or `False`] tells `IBDSim` to consider (1) time constant lattice size if set to false; Or (2) a linear, exponential or logistic continuous change in lattice bewteen Gn_X and Gn_{X+1}. This option works like the `Continuous_Deme_Size_Variation` option describe above. Note that this option requires a later `NewDemographicPhaseAt` to specify the ancient density at Gn_{X+1}.

- `Lattice_Logistic_Growth_Rate=0` is used with `Continuous_Lattice_Size_Variation=Logistic` to specify the growth rate of the logistic.

- `ForwardBarrier=False` [or `True`] tells `IBDSim` to simulate a linear barrier to gene flow. In the current version of IBDSim, the barrier can only be vertical or horizontal but not diagonal (i.e. either `X1_barrier==X2_barrier` or `Y1_barrier==Y2_barrier`). The difference between this option and the option below `BackwardBarrier` are explained below. The coordinates and the rate at which genes can cross the barrier are defined using the option below.

- `BackwardBarrier` or `Barrier =False` [or `True`] tells `IBDSim` to simulate a linear barrier to gene flow. In the current version of IBDSim, the barrier can only be vertical or horizontal but not diagonal (i.e. either `X1_barrier==X2_barrier` or `Y1_barrier==Y2_barrier`). The `BackwardBarrier` is a fast approximation using the barrier settings on backward migration events (see Sections 3.2 and especially 3.2.3 for details about backward and forward migration). Instead, option `ForwardBarrier=T` described below can be used to simulate a real barrier acting on forward dispersal. If the barrier dimension is the same as one dimension of the lattice (i.e. impossible to circumvent), then there is no difference between the two options. Differences will be noticeable only if the dimension of the barrier is very small compared to the maximum dispersal distance (`Dist_Max`). The coordinates and the rate at which genes can cross the barrier are defined using the option below.

- `X1_Barrier=1` [or any positive integer $<$ `Lattice_SizeX`] is the X coordinate of the lower point of the barrier.

- `X2_Barrier=1` [or any positive integer $<$ `Lattice_SizeX` and $>$ `X1_Barrier`] is the X coordinate of the upper point of the barrier.

- `Y1_Barrier=1` [or any positive integer < `Lattice_SizeY`] is the Y coordinate of the lower point of the barrier.
- `Y2_Barrier=1` [or any positive integer < `Lattice_SizeY` and > `Y1_Barrier`] is the Y coordinate of the lower point of the barrier.
- `Barrier_Crossing_Rate=0` [or any number between 0 and 1] is the rate at which genes cross the barrier at each generation.

4.3 Output files

`IBDSim` can generate different types of output files depending on the options chosen:

1. all simulated data sets in three different formats:

   (1) the extended input file format of `Genepop` v.4 (Rousset, 2008) with spatial coordinates of sampled individuals,

   (2) the format of `Geneland` (Guillot *et al.*, 2005) that consist in two files, one with the individual genotypes and one with their geographic coordinates,

   and (3) one specific file format that can be read as input for MIGRATE (Beerli & Felsenstein, 2001).

   For models dealing with sequence data (*i.e.* JC69, K80/K2P, F81, HKY85 and TN93 models), `IBDSim` can also generate NEXUS files containing the sample haplotypes and/or the sequences of all the genes in the simulated sample. See data file output options p.27 for a detailed description of each of those formats.

2. a summary file named `"simul_pars.txt"` where most parameter values used for the simulation are summarized and some statistics of the chosen dispersal distribution are computed (mean dispersal, second moment σ and kurtosis, etc...).

3. a summary statistic file named `"Various_Statistics.txt"` where the average over all multilocus runs of various genetic statistics, such as TMRCA, probability of identity between pairs of genes, observed and expected heterozygosity (Nei, 1987), variance in allelic size, Garza and Williamson's M statistic (Garza & Williamson, 2001), F_{IS} and mean coalescence times are computed on the simulated data sets as well as theoretical expectations, based on mutation rates, populations sizes and number of possible allelic states for some of those statistic in models where simple analytical results are available.

4. a file named `"Iterative_Statistics.txt"` with all records, for each simulated data file with details for each locus and average values among loci of various genetic statistics (observed and expected heterozygosity, variance in allelic size, Garza and Williamson's M statistic, F_{IS} and number of alleles). This file is presented as a table with the first line containing the names of each column (i.e. each statistic, usually straightforward to understand) followed by one line per simulated data set with the corresponding values. Note that it has specific values for each loci as well as averages for all loci for each data set (i.e. for each line) so that each statistic is represented by `Locus_Number` + 1 columns.

5. a file named `"Iterative_IdProb.txt"` with frequencies of pairs of genes identical in state at all distances represented in the sample. Each line of this file represents one simulated data set and identity probability values are given in two columns as specific values for the simulated data set considered as well as average values considering all previous simulated data sets.

6. A file named `"Matrix_IdProb.txt"` with the average over all runs of probabilities of identities between pairs of genes computed on the generated data sets as a function of the location (i.e. spatial coordinates) of the genes on the lattice.

7. For each data set, a file named `"EmpDisp_DataFileName"` with the empirical effective dispersal distribution computed from all dispersal events that occurred during the multilocus coalescent trees used for the simulations. The dispersal distribution is represented as a table that can be used to plot an histogram. At the end of the file various statistics (mean, second moment σ, kurtosis and skewness) are computed on the whole distribution and on the semi distribution (axial values). Empirical dispersal distances are always computed considering the habitat as a plane, even if the simulation settings actually considers a torus. Discrepancies between theoretical and empirical dispersal distributions are thus expected with all edge effects, especially for small size lattices and/or large maximal dispersal distances.

8. A file named `"MeanEmpDisp.txt"` with the mean empirical effective dispersal distribution over all simulated data sets and, at the end of the file, various statistics (mean, second moment σ, kurtosis and skewness) are computed on the whole distribution and on the semi distribution (axial values). The format is the same as for the previous output file `"EmpDisp_DataFileName"`.

4.4 Interaction with Genepop

Interaction of IBDSim with Genepop to evaluate the performance of inferences under isolation by distance has been greatly enhanced in the latest version of Genepop (V. 4 and later). Genepop's behavior can now be controlled using an option file and by inline arguments in a console command line.
This allows batch calls to Genepop and repetitive use of Genepop on simulated data. Such automatic batch mode of Genepop makes it easy for anyone to test the performance of the regression estimators of $D\sigma^2$ by the regression methods (Rousset, 1997; Rousset, 2000; see the Genepop documentation section 5 for details), including the performance of the bootstrap confidence intervals, using simulated data sets produced by IBDSim. For example, users can easily evaluate the performance of two different estimators of the so-called neighborhood size under simulation conditions of their choice, by simulating samples using IBDSim and analyzing them using the Performance setting of Genepop V.4.

4.5 Interaction with Migraine

Interaction of IBDSim with Migraine to evaluate the performance of inferences under isolation by distance or a model of a single population with past variation in population size (OnePopVarSize, OPVS) is relatively easy and will be further enhanced in the future version of IBDSim. In its current state, IBDSim produces for each run (i.e. for say, 100 multilocus data sets simulated under a given scenario) a Migraine.txt file that can be used by Migraine as an input settings file to make demographic inferences on the simulated data sets. Such interaction between IBDSim and Migraine is currently restricted to IBD scenarios with allelic data types or to analysis under a single population model with past variation in population size (i.e. OnePopVarSize model in Migraine) with allelic or sequence data, but only with single marker analysis for the moment. Users who want to use Migraine on IBDSim simulated data sets are advised to carefully read the Migraine documentation.

5 Credits and Copyright (code, grants, etc.)

IBDSim uses R.J. Wagner's implementation of the Mersenne Twister random number generator, http://www-personal.umich.edu/~wagnerr/ and extracts of the "Mathlib: a C Library of Special Functions" code from the R foundation. IBDSim also uses small bits of code from Numerical Recipes in C by Press et al. This work was financially supported by the AIP project no. 02002 "biodiversité" from the Institut Français de Biodiversité and by

the Agence Nationale de la Recherche (EMILE NT09-611697 and IM-MODEL@CORAL.FISH projects).

Bibliography

Beerli, P. & Felsenstein, J., 2001. Maximum likelihood estimation of a migration matrix and effective population sizes in n subpopulations by using a coalescent approach. *Proc. Natl. Acad. Sci. U. S. A.* **98**: 4563–4568.

Chesson, P. & Lee, C. T., 2005. Families of discrete kernels for modeling dispersal. *Theor. Popul. Biol.* **67**: 241–256.

Cornuet, J.-M., Pudlo, P., Veyssier, J., Dehne-Garcia, A., Gautier, M., Leblois, R., Marin, J.-M. & Estoup, A., 2014. DIYABC v2.0: a software to make approximate Bayesian computation inferences about population history using single nucleotide polymorphism, DNA sequence and microsatellite data. *Bioinformatics* .

Crow, J. F. & Kimura, M., 1970. *An introduction to population genetics theory*. Harper & Row, New York.

Di Rienzo, A., Peterson, A. C., Garza, J. C., Valdes, A. M., Slatkin, M. & Freimer, N. B., 1994. Mutational processes of simple-sequence repeat loci in human populations. *Proc. Natl. Acad. Sci. U. S. A.* **91**: 3166–3170.

Endler, J. A., 1977. *Geographical variation, speciation, and clines*. Princeton University Press, Princeton.

Felsenstein, J., 1981. Evolutionary trees from DNA sequences: a maximum likelihood approach. *J. Mol. Evol.* **17**: 368–376.

Garza, J. C. & Williamson, E. G., 2001. Detection of reduction in population size using data from microsatellite loci. *Mol. Ecol.* **10**: 305–318.

Guillot, G., Mortier, F. & Estoup, A., 2005. Geneland: A program for landscape genetics. *Molecular Ecology Notes* **5**: 712–715.

Hasegawa, Kishino & Yano, 1985. Dating of human-ape splitting by a molecular clock of mitochondrial DNA. *J. Mol. Evol.* **22**: 160–174.

Hudson, R. R., 1993. The how and why of generating gene genealogies. In: *Mechanisms of molecular evolution* (N. Takahata & A. G. Clark, eds.), pp. 23–36. Sinauer, Sunderland, MA.

Jukes, T. & Cantor, C., 1969. Evolution of Protein Molecules. In: *Mammalian Protein Metabolism* (M. Munro, ed.), pp. 21–132. New York. Academic Press.

Kimura, M., 1969. The number of heterozygous nucleotide sites maintained in a finite population due to steady flux of mutations. *Genetics* **61**: 893–903.

Kimura, M., 1980. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotides sequences. *J. Mol. Evol.* **16**: 111–120.

Kimura, M. & Crow, J. F., 1964. The number of alleles that can be maintained in a finite population. *Genetics* **49**: 725–738.

Kingman, J. F. C., 1982. The coalescent. *Stoch. Processes Applic.* **13**: 235–248.

Kot, M., Lewis, M. A. & van den Driessche, P., 1996. Dispersal data and the spread of invading organisms. *Ecology* **77**: 2027–2042.

Leblois, R., Estoup, A. & Rousset, F., 2003. Influence of mutational and sampling factors on the estimation of demographic parameters in a "continuous" population under isolation by distance. *Mol. Biol. Evol.* **20**: 491–502.

Leblois, R., Estoup, A. & Rousset, F., 2009. IBDSim: A computer program to simulate genotypic data under isolation by distance. *MolecolRes* **9**: 107–109.

Leblois, R., Estoup, A. & Streiff, R., 2006. Habitat contraction and reduction in population size: Does isolation by distance matter? *Mol. Ecol.* **15**: 3601–3615.

Leblois, R., Pudlo, P., Néron, J., Bertaux, F., Beeravolu, C. R., Vitalis, R. & Rousset, F., 2014. Maximum likelihood inference of population size contractions from microsatellite data. *Mol. Biol. Evol.* **31**: 2805–2823.

Leblois, R., Rousset, F. & Estoup, A., 2004. Influence of spatial and temporal heterogeneities on the estimation of demographic parameters in a continuous population using individual microsatellite data. *Genetics* **166**: 1081–1092.

Malécot, G., 1975. Heterozygosity and relationship in regularly subdivided populations. *Theor. Popul. Biol.* **8**: 212–241.

Nei, M., 1987. *Molecular Evolutionary Genetics*. Columbia University Press, New York.

Nordborg, M., 2007. Coalescent theory. In: *Handbook of statistical genetics* (D. J. Balding, M. Bishop & C. Cannings, eds.), pp. 843–877. Wiley, Chichester, U.K., 3rd edn.

Ohta, T. & Kimura, M., 1973. A model of mutation appropriate to estimate the number of electrophoretically detectable alleles in a finite population. *Genet. Res.* **22**: 201–204.

Patil, G. P. & Joshi, S. W., 1968. *A dictionary and bibliography of discrete distributions*. Oliver & Boyd, Edinburgh.

Pritchard, J. K., Seielstad, M. T., Perez-Lezaun, A. & Feldman, M. W., 1999. Population growth of human Y chromosome microsatellites. *Mol. Biol. Evol.* **16**: 1791–1798.

Robledo-Arnuncio, J. J. & Rousset, F., 2010. Isolation by distance in a continuous population under stochastic demographic fluctuations. *J. Evol. Biol.* **23**: 53–71.

Rousset, F., 1996. Equilibrium values of measures of population subdivision for stepwise mutation processes. *Genetics* **142**: 1357–1362.

Rousset, F., 1997. Genetic Differentiation and Estimation of Gene Flow from FStatisticsUnder Isolation by Distance. *Genetics* **145**: 1219–1228.

Rousset, F., 2000. Genetic differentiation between individuals. *J. Evol. Biol.* **13**: 58–62.

Rousset, F., 2008. GENEPOP007: a complete re-implementation of the GENEPOP software for Windows and Linux. *Molecular Ecology Resources* **8**: 103–106.

Rousset, F. & Leblois, R., 2007. Likelihood and approximate likelihood analyses of genetic structure in a linear habitat: performance and robustness to model mis-specification. *Mol. Biol. Evol.* **24**: 2730–2745.

Rousset, F. & Leblois, R., 2012. Likelihood-based inferences under a coalescent model of isolation by distance: two-dimensional habitats and confidence intervals. *Mol. Biol. Evol.* **29**: 957–973.

Tamura, K. & Nei, M., 1993. Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Mol. Biol. Evol.* **10**: 512–526.

Vitalis, R., 2002. Sex-specific genetic differentiation and coalescence times: estimating sex-biased dispersal rates. *Molecular Ecology* **11**: 125–138.

Watts, P. C., Rousset, F., Saccheri, I. J., Leblois, R., Kemp, S. J. & Thompson, D. J., 2007. Compatible genetic and ecological estimates of dispersal rates in insect (*Coenagrion mercuriale*: Odonata: Zygoptera) populations: analysis of 'neighbourhood size' using a more precise estimator. *Mol. Ecol.* **16**: 737–751.

Index